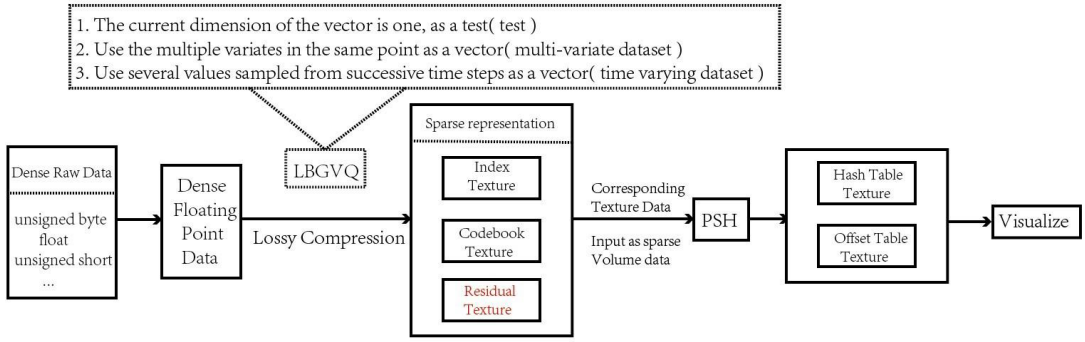
Weekly report

Introduction

This week, I mainly focused on improving the compression algorithm and testing different datasets to acquire the feedback. First, based on the basic implementation, I obtained and converted some datasets from volvis.org and climate datasets. A table includes all the related arguments and results of analysis is depicted. In order to achieve a better compress ratio, some methods have been employed. Index table(or index volume) is compressed via vector quantization procedure while the result shows it is no use to do such a work. Then, the dimension of the vector is increased, from 1 to something like 4 or 8, to make it a real vector quantization. However, the results are severely dataset-dependent. So I have to apply the last scenario, to change the data type of codebook, from unsigned char to unsigned short. By doing this, we can enlarge the size of the codebook so that the accuracy lost in the previous step(increase dimension) can be got back.

Details

As usual, the pipeline is showed here as a reference.



I have done the whole pipeline basically and tested some datasets. These datasets have two origins. The one is www.volvis.org and the other is climate datasets. As for the prior, I converted them from the unsigned char data to floating point data. The latter is extracted by wgrid.exe from the climate dataset. I set the dimension of the vector to be 1 and the size of codebook is 256, which means the type of the index volume is unsigned char. We have the results as the following table shows to us. Here, I define the residual as below

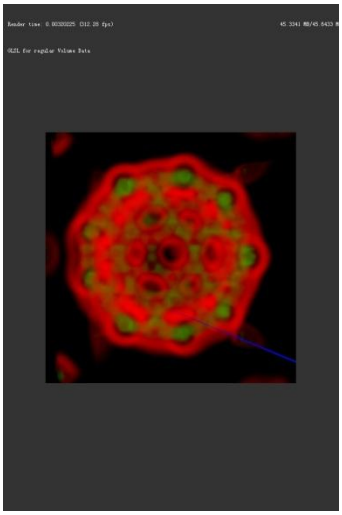
Residual = originalScalarValue – decodedScalarValue.

The first four datasets are from the website mentioned above and the rest are from climate datasets.

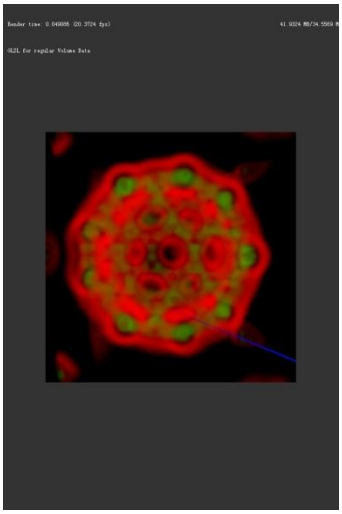
dataset	size	Data type	The proportion of voxels(abs of residual <	The proportion of voxels(abs of residual <	Max value of residual	Min value of residual	Size of codebook	Index table volume sparse?

			0.1)	0.00001)				
FBucky	32x32x32	FLOAT	94.85%	94.85%	5.5556	-2.444	256	
Fneghip	64x64x64	FLOAT	96.806%	96.806%	0.6538	-0.6258	256	
FhydrogenAtom128_128_128	128x128x128	FLOAT	99.9259%	99.9259%	12.8079	-1.33333	256	
Fbonsai256_256_256	256x256x256	FLOAT	97.2944%	97.2923%	1.64781	-1.57593	256	
CLWMR360_181_21	360x181x21	FLOAT	100%	99.998%	5.075e-005	-2.3143e-005	256	
RH360_181_31	360x181x31	FLOAT	100%	100%	0	0	256	
ABSV360_181_26	360x181x26	FLOAT	100%	99.978%	4.55517e-005	-4.36514e-005	256	

We also have two pictures to show the results visually and it is clearly that the original volume and the decoded volume are acceptable similar by the results of both the table and the pictures.



Original volume rendering results



Decoded volume rendering results

Dataset: Fbucky.raw

By doing experiments, it is said that our method is capable for the relative clean datasets. If the noise is not very heavy or the scalar value variation is small or stable, we can obtain a sparse residual volume which can be employed in the following PSH stage. The compression ratio is showed below

Compression ratio = 0.25 + sizeofCodebook/sizeofOriginalVolumeData

The current sizeofCodebook = 256*4byte = 1KB， take the FBucky.raw as an example，its' size is 32768*4byte = 128KB，so sizeofCodebook/sizeofOriginalVolumeData = 0.0078125.

Improvements and results

In order to achieve a better compress ratio, some ideas have been employed. The first try is to compress the Index table(or index volume) via vector quantization procedure while the result shows it is no use to do such a work. It is because data type of the Index volume is unsigned char, we cannot decrease the size of the Index table by vector quantization in current case. Then, I changed the code mapping from one voxel to one voxel to block voxels to one voxel, this will decrease the size of Index table. In fact, the dimension of the vector is increased, from 1 to something like 4 or 8, to make it a real vector quantization. However, the results are severely dataset-dependent, only one dataset has a sparse residual volume till now. So I have to apply the last scenario, to change the data type of codebook, from unsigned char to unsigned short. By doing this, we can enlarge the size of the codebook so that the accuracy lost in the previous step(increase dimension) can be got back. It has been done and the issue now should be the performance. As the size goes on, the time needed to compute is obviously increased. I will keep improving next week.